ORACLE

SPARC T-SERIES

An Oracle Technical White Paper
April 2012

High Performance Security For Oracle Database and Fusion Middleware Applications using SPARC T4

Introduction

This document presents the high performance security characteristics of using the cryptographic acceleration capabilities of Oracle SPARC T4 processor based Servers for Oracle Fusion Middleware and Oracle Database applications. This document explores the technical details of Oracle SPARC T4 processor's high-performance cryptography features and its applied techniques to secure Oracle Weblogic server, Oracle Fusion Middleware and Oracle Database server based applications. In addition, it also drills down on the technical pre-requisites, configuration, and deployment and its verification guidelines for delivering Oracle SPARC T4 hardware-assisted cryptographic acceleration solutions for end-to-end application security scenarios where the use of cryptography is deemed critical. The derived high performance security benefits can be leveraged into different application solutions of Oracle "Red Stack" that represents the complete set of application software, middleware and infrastructure software.

Target Audience and Assumed Knowledge

This document is intended for security enthusiasts, developers and administrators of Oracle Weblogic server, Oracle Fusion Middleware, Oracle Database and Solaris deployed applications, who have been tasked to design, deploy and integrate the on-chip cryptographic capabilities of Oracle SPARC T4 processor based servers. The developers and administrators should be familiar with the installation of Oracle SPARC T4 processor based servers, Oracle Solaris 11, Oracle Database Advanced Security for Transparent Data Encryption features, Oracle WebLogic server and Fusion Middleware security techniques for secure communication using SSL/TLS protocols and secure XML Web services using WS-Security standards.

The Role of Oracle SPARC T4 Processor in Application Security

As security has taken unprecedented importance in all facets of the IT industry, today organizations are proactively adopting to cryptographic mechanisms to protect their business information from internal and external threats, unauthorized access and also to ensure its confidentiality and integrity during transit and storage. Cryptographic operations are heavily compute-intensive which burdens the host system with additional CPU cycles and network bandwidth resulting significant degradation of overall throughput of the system and its hosted applications. For example, a host server capable of processing 1000 transactions per second can perform only 10 transactions per sec after deploying SSL for securing the hosted application. To speed up cryptographic performance, security experts often recommend and use cryptographic accelerator appliances to offload cryptographic operations and save CPU cycles for enhancing the system throughput and its hosted applications. While useful, adopting a specialized appliance for offloading cryptographic operations introduces a new set of complexities and issues in terms of additional installation, configuration and testing procedures that significantly increases the power demands and costs of deployment projects. Foreseeing the need for special-purpose hardware that can outpace workload demands, Oracle introduced the industry's fastest on-chip hardware cryptographic capabilities into its family of Oracle SPARC T-series processors - UltraSPARC® T1, T2, T2 Plus, T3 and T4 processors equipped with CoolThreads™ technology. The following graph (Refer Figure 1) presents the security performance gains achieved by the Oracle SPARC T4 processor based on-core cryptographic instructions (hardware acceleration) for a Web application security (SSL/TLS communication scenario) in comparison with just using software-managed cryptographic operations (no acceleration).

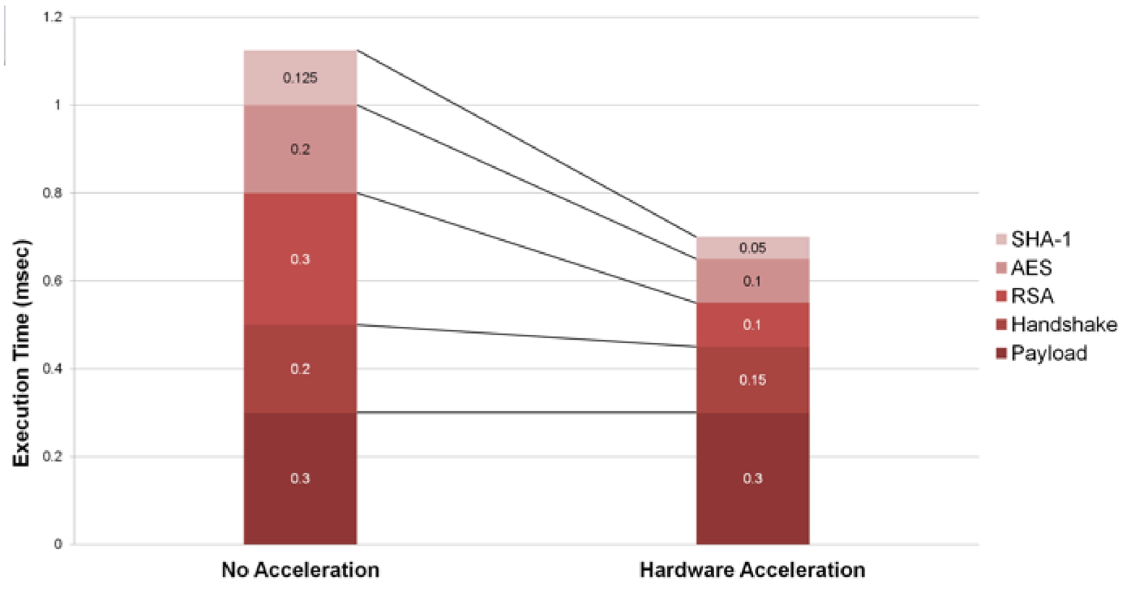


Figure 1: Web Application Security: No Acceleration vs. SPARC T4 Hardware Acceleration

Oracle SPARC T4 – Integrated Cryptographic Acceleration

The Oracle’s new SPARC T4 processor is the fifth generation of Oracle SPARC T-series processors family that represents a fundamental redesign of the core within a SPARC multi-core/multi-threaded processor architecture. By redesigning the cores within each processor, introducing a new floating-point pipeline and further increasing network bandwidth, the new T4 processor is able to provide approximately 5X the single-threaded throughput gains comparing to its predecessor SPARC T3 processor. The T4 processor provides 64 threads per processor, on-chip memory management, two 10 GbE interfaces, dual on-chip based PCIe Generation 2 root complexes, on-chip/on-core based cryptographic acceleration and hardware-enabled virtualization capabilities. As a result, The SPARC T4 processor eliminates the need for expensive custom hardware and software development by integrating computing, security, and I/O onto a single chip.

The SPARC T4 processor is shown in Figure 2.

Figure 2: Oracle SPARC T4 processor

The Oracle SPARC T4 features on-core cryptographic algorithms made available as unprivileged ISA instructions. To support cryptographic operations, each core of the Oracle SPARC T4 processor contains a Stream Processing Unit (SPU) that performs cryptographic functions at the same clock speed as the core. The SPU on each is implemented within the core pipelines, accessible by 29 new user-level instructions for performing cryptographic functions. During a cryptographic operation, the cryptographic function will leverage SPU and also use parts of Floating Point/Graphics Unit (FGU) and Integer Execution Unit (EXU) pipelines with Floating-pont Register Files (FRF) and Integer Register Files (IRF). The logical depiction of SPU in Oracle SPARC T4 processor is shown in Figure 3. As a result, the SPU is designed to achieve wire-speed encryption and decryption on the processor's 10 GbE ports.

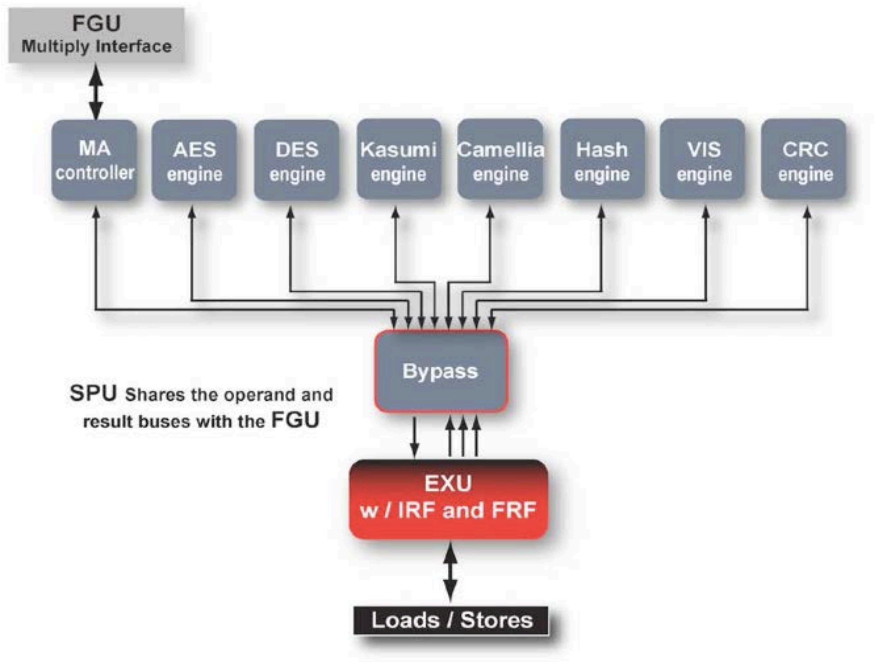


Figure 3: Oracle SPARC T4 processor – Logical Depiction of Stream Processing Unit (SPU)

The following table shows the cryptographic algorithms supported by the Oracle SPARC Enterprise T-series processors (Table 1).

ON-CORE CRYPTOGRAPHY SUPPORT	**ORACLE SPARC T4**
ACCELERATOR DRIVER	Userland (No drivers required)
PUBLIC KEY ENCRYPTION	RSA, DSA, DH, ECC
BULK ENCRYPTION	AES, DES, 3DES, RC4, Kasumi, Camellia
MESSAGE DIGESTS	CRC32c, MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512
APIS	PKCS#11 Standard Ucrypto APIs

Table 1: Oracle SPARC T4 Processor - Supported Cryptographic Algorithms

While comparison with Intel AES-NI based processors with Oracle SPARC T4 offers comprehensive set of algorithms supporting long-list of public-key encryption, Symmetric-key encryption and message-digest algorithms.

Cryptographic Operational Models: SPARC T4 versus SPARC T3

By providing on-core cryptographic instructions using SPU, the SPARC T4 processors provide significant cryptographic performance gains to userland applications than earlier processors such as the SPARC T3 and SPARC T2 Plus processors. The SPARC T3 and T2 Plus processors provided dedicated cryptographic accelerators (co-processors) accessible only via accelerator drivers and managed by the Solaris kernel, which caused startup overheads (due to kernel context switching and buffer copying) on select application payloads particularly using smaller packets (less than ~1k). With SPARC T4 processor, the userland applications can directly access on-core based cryptographic algorithm functions and automatically perform cryptographic functions in hardware - without requiring use of special drivers, kernel environment and root permissions.

The following Figure - 4 represents the comparison of operational models of Oracle SPARC T4 and Oracle SPARC T3 / T2 Plus processors.

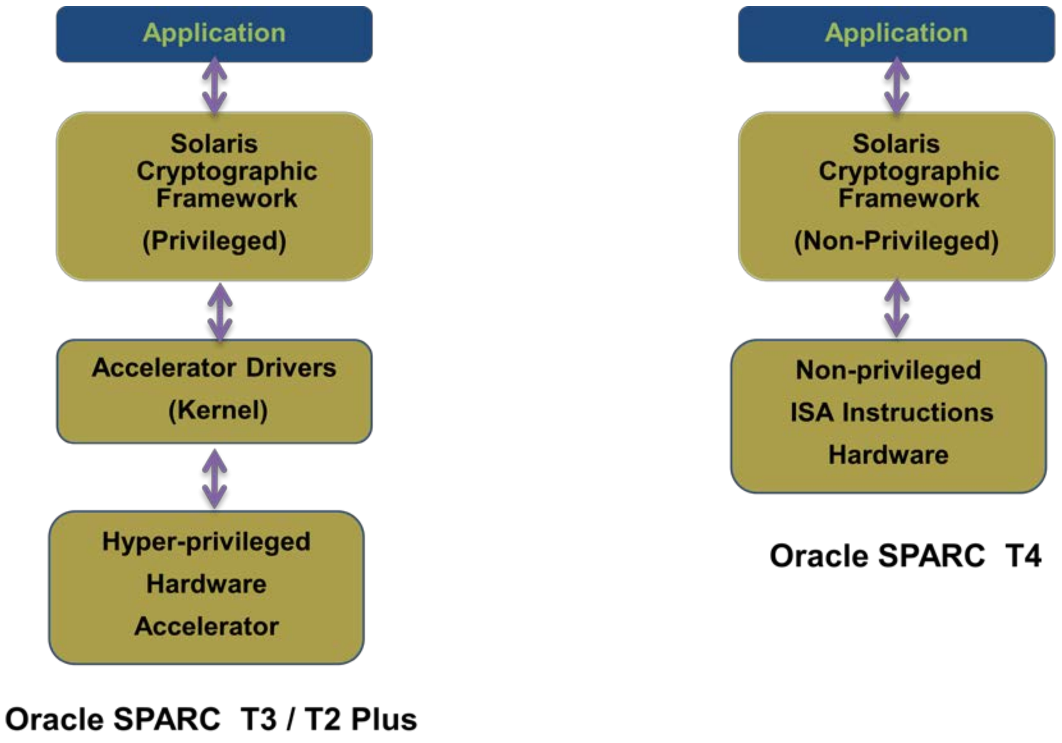


Figure 4: Operational Model Comparison - SPARC T3/T2 Plus and SPARC T4

The Role of Oracle Solaris Cryptographic Framework

In practice, the Oracle Solaris Cryptographic Framework acts as the core intermediary between the applications and the underlying hardware. The framework enables user-level applications to automatically off-load cryptographic operations to hardware-assisted cryptographic acceleration. The Oracle Solaris Cryptographic Framework libraries provides a set of cryptographic services and API support whereby both kernel and user-level application consumers can transparently delegate the cryptographic operations to hardware - without adding any new code to the application at all.

Oracle SPARC T4: High Performance Security Characteristics

To better understand the Oracle SPARC T4 hardware-assisted cryptographic capabilities, let's begin with examining its performance characteristics for the applied security scenarios of Oracle Fusion middleware and Oracle database based applications.

Hardware and Software Environments Used

The following Oracle SPARC T4 hardware and Oracle application software environment was used for testing and validating the performance characteristics (Refer Table 2):

OPERATING SYSTEM	HARDWARE ENVIRONMENT	APPLICATIONS
Oracle Solaris 11	• Oracle SPARC T4-4 Server • Oracle SPARC T4-2 Server	Oracle WebLogic Server 10.3.6.0 64-bit for Solaris SPARC Oracle Fusion Middleware 11.1.1.6.0 Oracle Database 11.2.0.3 Oracle Enterprise Content Management 11.1.1.6

TABLE 2: HARDWARE AND SOFTWARE ENVIRONMENT USED

Oracle WebLogic Security – SSL/TLS Performance

The following graph (Refer Figure 5) represents the SSL operations performance characteristics of the following Oracle WebLogic 10.3.6 Server deployed with Oracle Enterprise Content Management 11.1.1.6 application. The test environment is as follows:

Machine used:
T4-4 Server, 64 Gb Memory

Software configuration:
Oracle WebLogic 10.3.6
Solaris 11
JDK 1.6u26, JDK7

No. of Concurrent users:
350

RSA Keysize used:
1024 bit, 2048 bit

JCE enforced SSL ciphersuites:
TLS_RSA_WITH_AES_128_CBC_SHA

The security scenarios are as follows

a. WebLogic Managed Server configured to use No SSL

b. WebLogic Managed Server SSL listener configured with JKS keystore – Using default Certicom SSL provider configuration.

c. WebLogic Managed Server SSL listener configured with JKS keystore – JSSE configuration (Cryptographic operations delegated to T4 via SunPKCS11 provider).

d. WebLogic Managed Server SSL listener configured with JKS keystore –(Cryptographic operations delegated to T4 via Userland Crypto provider – Oracle Ucrypto provider).

Faban was used as the load driver for deriving SSL performance, simulating 350 concurrent users for this test.

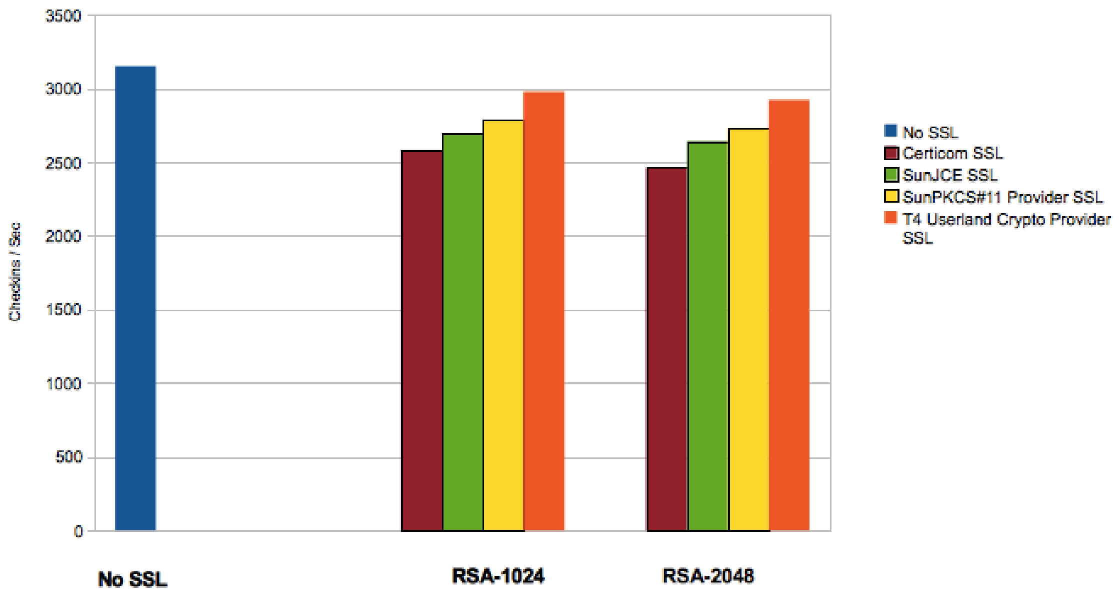


Figure 5: WebLogic SSL Performance Characteristics on Oracle SPARC T4-4 Server

As a result of using Oracle SPARC T4 on-core cryptographic instructions for Weblogic SSL scenarios solidly delivered the overall application performance with SSL showing 4% - 5% overhead for RSA-1024 based certificates and 5% - 6% overhead for RSA-2048 certificates while comparing to overall application performance with No SSL. The results showed only a negligible difference between RSA-1024 versus RSA-2048 and between the unsecured applications versus on-core cryptographic accelerated solution.

Oracle Fusion Middleware Security – WS-Security Performance

The following graph (Refer Figure 6) represents the comparison of end-to-end overall application throughput performance including both SSL and WS-Security operations using hardware-assisted cryptographic acceleration and without using them. The tests ran a Java EE/JAX-WS Web Services application (sample application bundled with WebLogic 10.3.6) using a 500k XML message payload with the addition of ~800k SOAP binary attachment payload deployed on Oracle WebLogic server 10.3.6 running on Oracle SPARC Enterprise T4-1 server. The XML Web service is configured with an Oracle Fusion Middleware 11.1.1.6 (Oracle Web Services Manager) enforced policy (`wss11_username_token_with_message_protection`) using Algorithm suite `Basic256` (AES-256 for message encryption) and the Oracle WebLogic Server configured to SSL/TLS cipher suite TLS_RSA_WITH_AES_128_CBC_SHA.

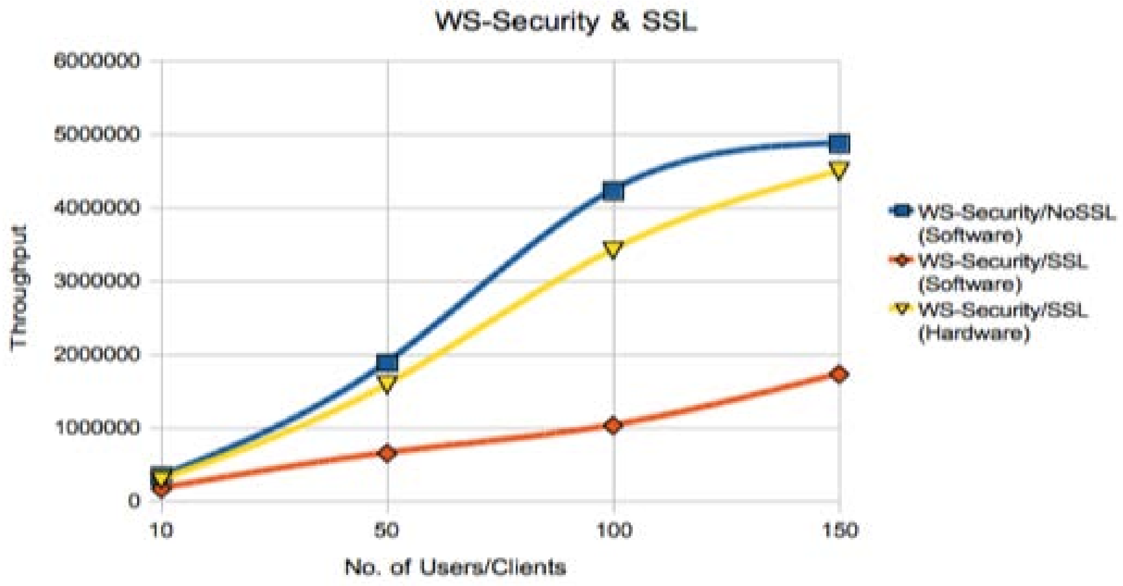


Figure 6: SSL and WS-Security Combined – Performance

The results showed using Oracle SPARC T4 hardware-assisted cryptographic acceleration (enabled via Oracle UCrypto and SunPKCS11 providers) significantly contributed to combined SSL/TLS and WS-Security scenarios delivering more than **200% of overall application performance gain for Two-way SSL and WS-Security operations** in comparison with SSL/WS-Security operations using software .

Oracle Database Security – TDE Performance

The following graph (Refer Figure 7) represents the comparison of relative and absolute latencies (units in microsec) of tablespace encryption between Software and Oracle SPARC T4 hardware-assisted cryptography. The tests were conducted with 8K data block size with AES-CFB 192-bit encryption using Oracle SPARC T4-2 server running Oracle 11gR2 (11.2.0.3) on Solaris 11.

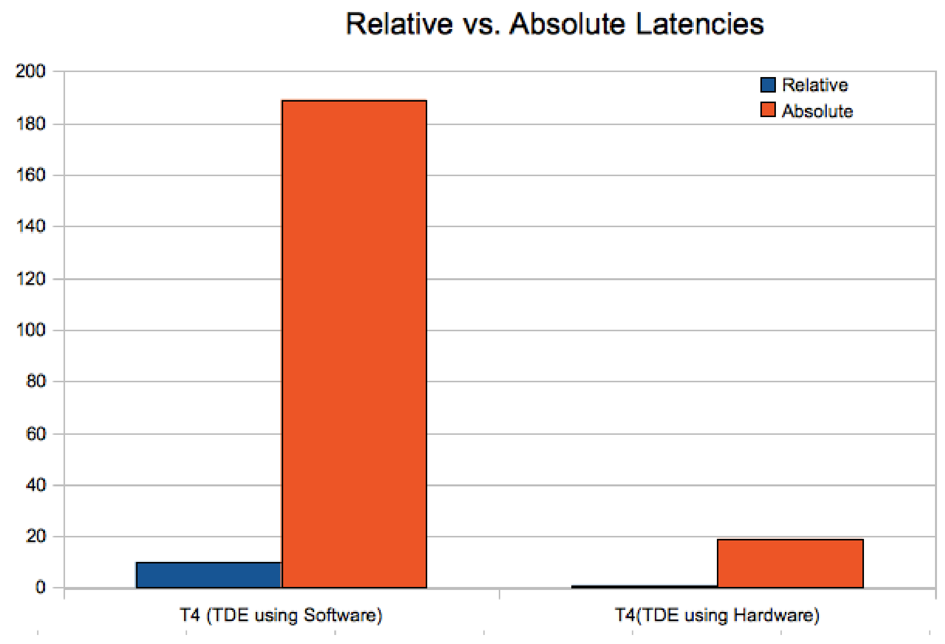


Figure 7: Comparison of Relative and Absolute latencies: Encryption of Unit Data on Oracle SPARC T4-2

The following graph (Refer Figure 8) comparison of relative time taken (units in microsec) for tablespace encryption using mechanisms – Oracle SPARC T4 hardware-assisted vs. Software.

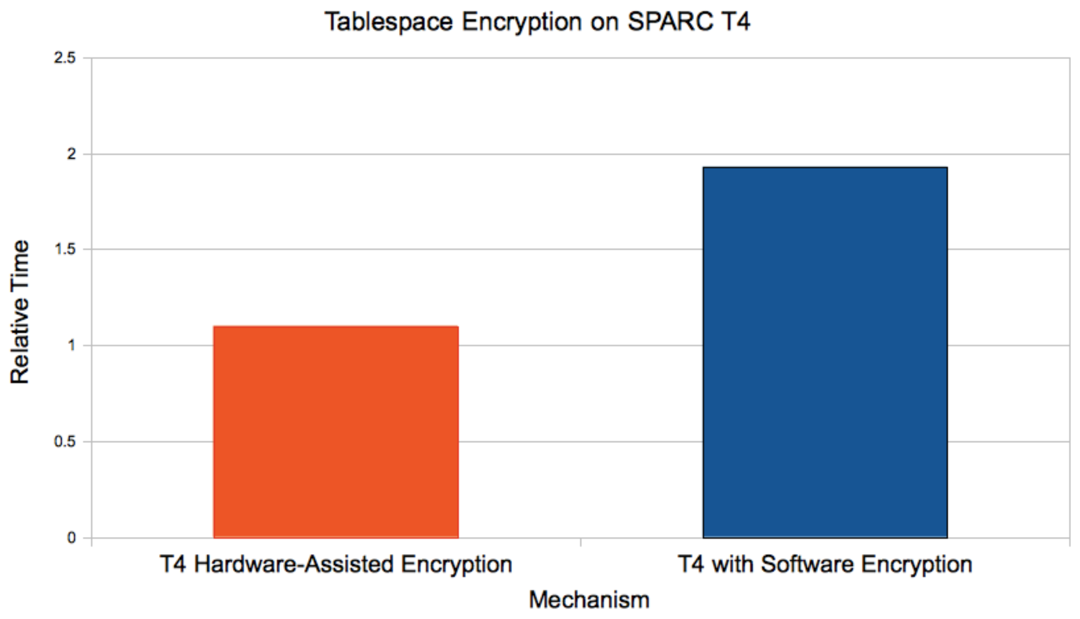


Figure 8: Comparison of Relative time: Oracle TDE on Oracle SPARC T4-2

The following graph (Figure 9) represents the comparison of relative latencies (units in microsec) of decryption of unit data between Software and Oracle SPARC T4 hardware-assisted cryptography.

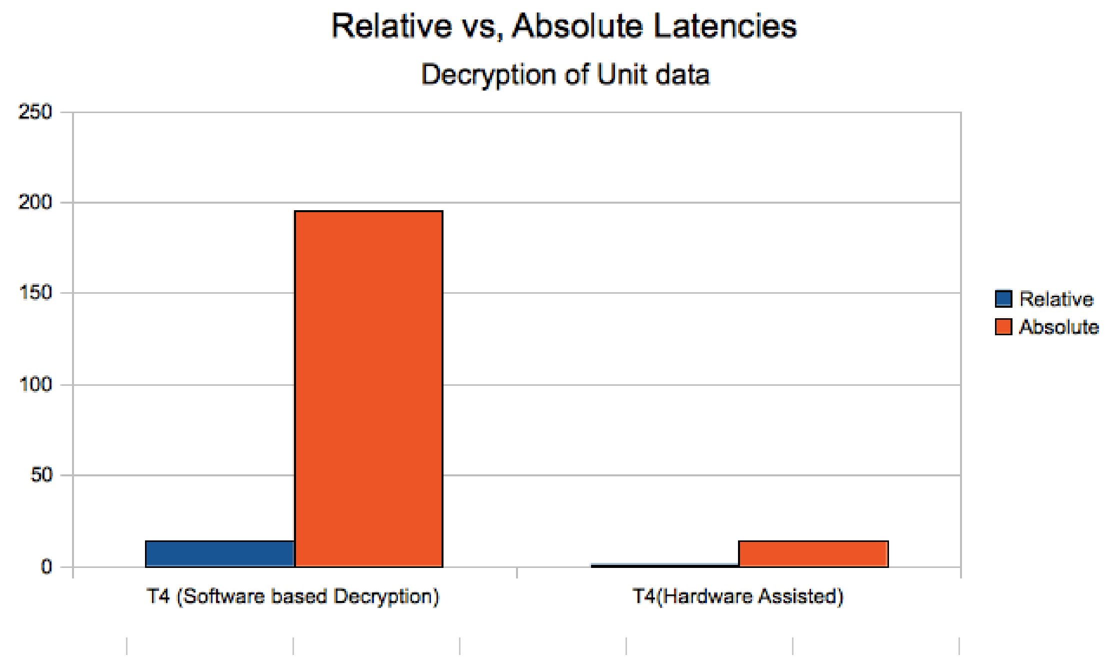


Figure 9: Comparison of Relative and Absolute latencies: Decryption of Unit Data on Oracle SPARC T4-2

Key observations from Oracle TDE on Oracle SPARC T4-2 :

- Only about 10% overhead for Encryption using hardware-assisted cryptographic acceleration on T4.
- Oracle SPARC T4 hardware assisted cryptographic acceleration is about 42% faster than software implemented encryption.
- Pure cryptographic operation latency overhead for a single encrypted block access is only about 7% over that of cleartext.
- Using hardware acceleration, decryption overhead on T4 is lesser than similar overhead on state of the art X86 hardware.
- From the Oracle SPARC SuperCluster deployment standpoint, the added advantage is interoperability - data encrypted on database running Oracle SPARC T4 server can be decrypted on X86 storage cells.

In the following sections, we will explore on the technical details of Oracle SPARC T4 hardware-assisted cryptographic capabilities can be leveraged into applied security scenarios of Oracle Fusion middleware and Oracle database applications.

Enabling Cryptographic Acceleration For End to End Security

In a typical deployment scenario of Oracle Fusion Middleware and Oracle Database environment enabling an end-to-end security topology (Refer Figure 10) requires the use of encryption at all levels to ensure secure data in transit, secure data in processing and to secure data in storage. The Oracle T4 based cryptographic acceleration can significantly contribute to the end-to-end security topology where the use of cryptographic mechanisms is deemed critical. The delivery of high performance security is accomplished through Oracle Solaris Cryptographic Framework that allows applications can transparently offload and delegate their cryptographic operations to the on-core cryptographic capabilities of Oracle SPARC T4 processor. In addition with the support of Oracle Solaris Cryptographic Framework the applications can leverage the Oracle Solaris facilitated key storage and management features.

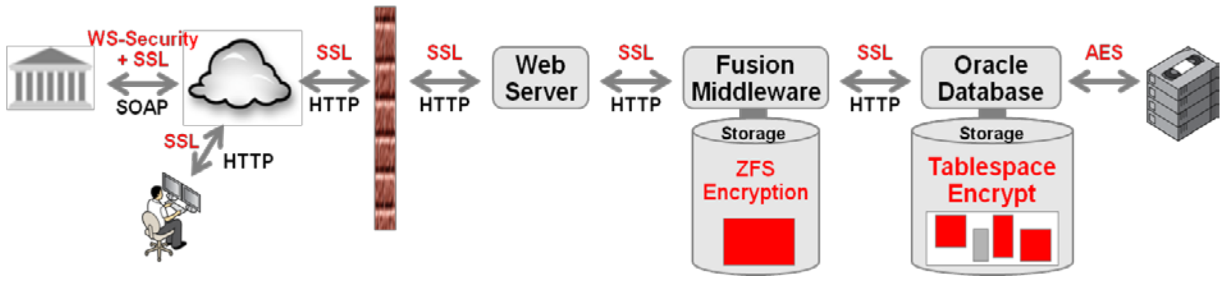


Figure 10: Deployment of Oracle Fusion Middleware and Database: End to End Security Topology

Oracle Fusion Middleware: Applied Security Scenarios

The Oracle SPARC T4 processor's on-core cryptographic acceleration capabilities can be accessed in a variety of ways by the Oracle WebLogic server, which host the Oracle Fusion Middleware application environment and its deployed security scenarios. The Oracle WebLogic server can offload select cryptographic operations for the following security scenarios.

Transport-layer Security

- SSL/TLS acceleration offloads computationally intensive public-key cryptographic operations such as RSA and ECC.
- RMI over IIOP with SSL uses SSL/TLS to protect IIOP connections to RMI remote objects.

Message-Layer Security

- Acceleration of cryptographic operations intended for supporting XML Web Services security standards such as WS-Security, WS-SecurityPolicy. XML Web services security relies on public-key encryption, digital signature (ex. RSA, DSA), bulk encryption (ex. AES, DES) and message digest (ex. SHA-1, SHA-256, MD5) functions intended for supporting XML encryption, XML digital signature and related cryptographic operations.

Secure Data at Rest

- Acceleration of cryptographic operations intended for supporting data stored in file system. This will be accomplished through the use of ZFS encrypted file system. The Solaris 11 based ZFS encryption automatically leverages T4 hardware-assisted cryptographic acceleration.

Transport-Layer Security Acceleration

To secure the transport layer, the Oracle Fusion Middleware applications and its hosting Oracle WebLogic server relies on the underlying Java Cryptographic Extensions (JCE) provider for its SSLv3/TLSv1 protocols. On Oracle Solaris 11 SPARC deployments, the Sun JCE provider is bundled with the Sun Java runtime environment facilitates the PKCS#11 interfaces for delegating the cryptographic operations intended for SSL. The availability of PKCS#11 interfaces via Java Cryptographic Extension (JCE) framework enables the WebLogic server deployed SOA applications, XML Web services and Java EE applications to automatically take advantage of on-core cryptographic acceleration for SSL based cryptographic workloads. With the release of JDK7 update 4, Oracle introduced Oracle Ucrypto provider, which provides a specialized interface bypassing PKCS11 and automatically leverages Oracle SPARC T4 hardware-asssisted cryptographic acceleration.

Alternatively, the WebLogic server can make use of Solaris KSSL as an SSL proxy for handling transport-layer security operations. Like any other kernel module, KSSL tightly integrates with the Solaris kernel and makes use of on-core cryptographic acceleration provided by Oracle SPARC T4 processor.

Using Sun JCE

By default, the Oracle WebLogic server on Oracle SPARC T-Series servers relies on the JDK and its Sun JCE provider environment for handling cryptographic operations. The Sun JCE provides a PKCS#11 implementation (Java SunPKCS11) that enables Java applications to access hardware cryptographic tokens – such as Secure Sockets Layer (SSL) accelerators, Smartcards, and HSMs. The SunPKCS11 provider is a Java based PKCS#11 implementation that integrates with underlying PKCS#11 implementations provided by the SCF and its exposed cryptographic providers. The SunPKCS11 provider does not implement its own cryptographic algorithms. With JDK7 update 4 and above, the JDK includes a new Oracle Ucrypto provider to support Oracle SPARC T4 processor's on-core cryptographic instructions (Refer Figure 11).

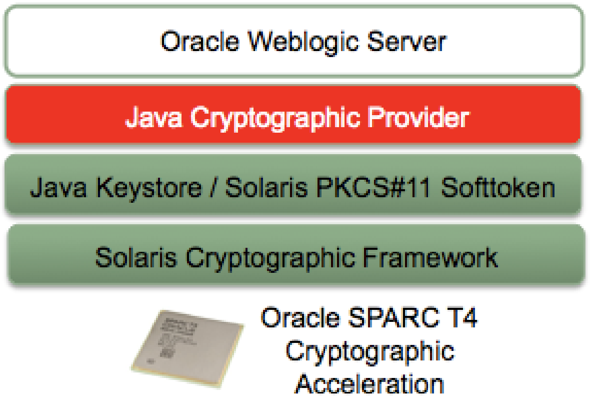

Oracle Weblogic Server
Java Cryptographic Provider
Java Keystore / Solaris PKCS#11 Softtoken
Solaris Cryptographic Framework
Oracle SPARC T4 Cryptographic Acceleration

Figure 11: Oracle WebLogic Server Security using Oracle SPARC T4 Server

As a pre-requisite, for leveraging hardware-assisted cryptography it is critical that the Oracle WebLogic SSL must be configured to use JSSE based SSL.

- To begin with, refer to all Weblogic SSL configuration pre-requisites including the setting up the keystore for storing SSL certificates. It is recommended to use Java keystore (JKS) which requires choosing option "**Custom Identity and Java Standard Trust"** as described in Oracle Weblogic Security documentation "Configure keystores".

- Using JSSE based SSL automatically leverages the SunPKCS11 provider implementation pre-configured with the Java runtime environment on Solaris SPARC and X64. To enable JSSE based SSL, refer to the steps as described in Oracle Weblogic Security documentation Using the JSSE-Based SSL Implementation.

Accelerating SSL using SunPKCS11 Provider

In a typical WebLogic server installation on Solaris, the Java runtime environment is pre-configured to make use of the SunPKCS11 provider. To verify this refer to the Java security properties file located at `$JAVA_HOME/jre/lib/security/java.security` properties file and make sure it identifies SunPKCS11 as the default provider.

```
security.provider.1=sun.security.pkcs11.SunPKCS11
                        ${java.home}/lib/security/sunpkcs11-solaris.cfg
```

The `$JAVA_HOME/jre/lib/security/sunpkcs11-solaris.cfg` file contains the configuration information used by the SunPKCS11 provider for accessing the SCF. To help Java applications benefit from cryptographic acceleration, the `sunpkcs11-solaris.cfg` file allows users to disable the soft-token (user-level provider) mechanisms and in turn delegate them to the underlying hardware-based cryptographic accelerator/token provider and the mechanisms.

The following steps explain how to configure Oracle WebLogic Server for SSL acceleration using the on-chip cryptographic acceleration capabilities of Sun UltraSPARC T-Series servers.

1. Configure WebLogic Server to listen for SSL. Before configuration, make sure you obtain the private keys, server certificate including the public key and trust CA certificates from a Certificate Authority (CA) and then store them into the Java keystore (Identity and Trust keystores) configured within the WebLogic server environment. In case of development and testing, you may choose to use a self-signed certificate, private key and trusted CA certificate created using Java key tool. Use the WebLogic Server Administration Console to configure the identity and trust keystores.

   Follow the SSL configuration guidelines specified in the *Oracle Fusion Middleware - Securing WebLogic Web Services for Oracle WebLogic Server 11g* guide.

2. Verify and confirm that the Oracle WebLogic Server is listening and responds over the SSL port.

3. Enable cryptographic acceleration by editing the Java SunPKCS11 provider configuration file located at `$weblogic-javahome/jre/lib/security/sunpkcs11-solaris.cfg`. This file contains vital attributes for allowing Oracle WebLogic server to access the cryptographic mechanisms and attributes supported by the underlying on-chip cryptographic accelerator, which can be enabled or disabled in the SunPKCS11 configuration file.

4. It is important to enable and enforce delegation of the required cryptographic mechanisms to the underlying PKCS#11 provider that facilitates the hardware accelerator support. Make sure to include the required public key cryptographic mechanisms (ex. `CKM_RSA_PKCS`) in the Java SunPKCS11 provider configuration file that lists as part of `enabledMechanisms` list or removes the mechanisms from the list of `disabledMechanisms` of the Java SunPKCS11 configuration file. Doing so forces the required public key operations to be performed by the hardware. Additionally, you may enable or disable the bulk encryption and message digest algorithms in the list that forces those operations performed by the hardware.

5. Restart the Oracle WebLogic server

Accelerating SSL using Oracle Ucrypto Provider

With the release of JDK7 update 4, Oracle introduced a new Ucrypto provider that leverages Solaris 11 Ucrypto APIs for offloading and delegating of cryptographic operations supported by the Oracle SPARC T4 based on-core cryptographic instructions. To leverage the Oracle Ucrypto provider, it is required to make sure that the Ucrypto provider is identified as the default provider in the Java security properties file `java.security` located at `$JAVA_HOME/jre/lib/security/` directory.

```
security.provider.1=com.oracle.security.ucrypto.UcryptoProvider
                        ${java.home}/lib/security/ucrypto-solaris.cfg
```

Using Solaris KSSL

KSSL is a Solaris kernel module that acts as a server-side SSL protocol for offloading operations such as SSL/TLS-based communication, SSL/TLS termination, and reverse proxies for end-user applications. KSSL takes advantage of the SCF to act as an SSL proxy server, performing complete SSL handshake processing in the Solaris OS kernel. KSSL automatically uses the Oracle SPARC T4 hardware-assisted cryptographic acceleration, PKCS#11 keystores, and Hardware Security Modules for enabling SSL acceleration and secure key storage (Refer Figure 12).

The key technology aspects and the security benefits of using KSSL include:

- Helps to introduce—non-intrusively—an SSL proxy server for Web servers, Java EE application servers, and applications that do not support SSL.
- Listens to secured requests on the designated SSL port (ex. http://:443) and renders cleartext traffic via a reverse proxy port (ex. http://:7001) for an underlying WebLogic application server or a load balancer that supports multiple instances of application servers (Oracle WebLogic Managed Servers). In a real world scenario, KSSL proxy can reside in the Solaris OS global zone and redirect to load balancer that performs round-robin delivery of requests/responses to a set of WebLogic managed servers running in non-global zones.
- All SSL operations, including the SSL handshake and session state, are performed asynchronously in the Solaris kernel and without the knowledge of the target application server. Automatically uses the Solaris Cryptographic Framework for off-loading operations to the underlying hardware cryptographic accelerators without any extra effort.
- Manages all SSL certificates independently and supports most standard formats, including PKCS12 and PEM. Key artifacts can be stored in a flat file (OpenSSL) or a PKCS#11 conforming keystore (ex. HSMs, NSS, Solaris PKCS#11 Sofftoken) to help ensure the protection of private keys.
- Supports the use of Solaris Zones. Each IP-identified zone can be configured with a KSSL proxy.
- Delivers 15% to 25% faster SSL performance compared to traditional SSL configurations used in popular Web servers and Java application servers, such as Oracle WebLogic Server and Oracle GlassFish application Server.

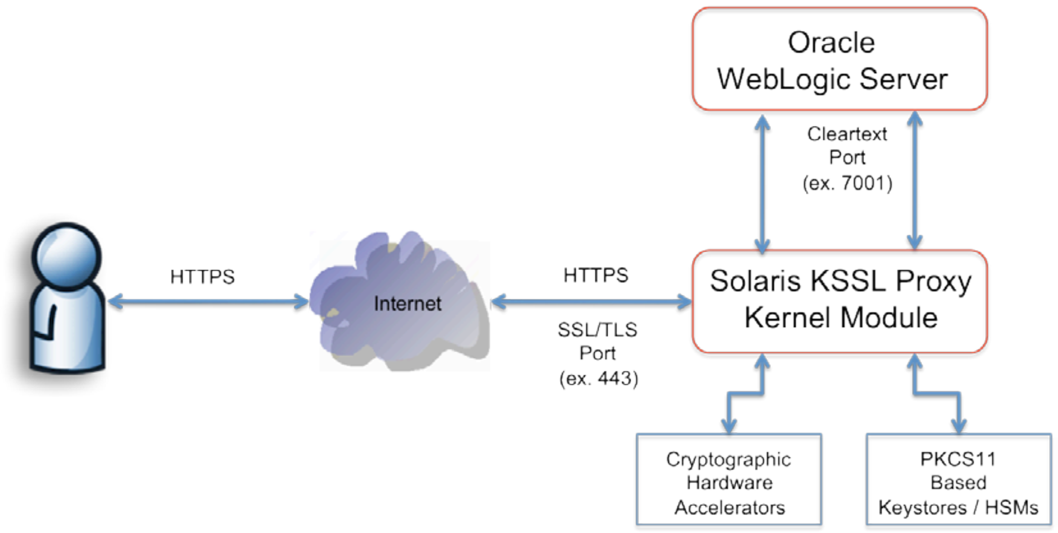

Oracle WebLogic Server
Cleartext Port (ex. 7001)
HTTPS
Internet
HTTPS
SSL/TLS Port (ex. 443)
Solaris KSSL Proxy Kernel Module
Cryptographic Hardware Accelerators
PKCS11 Based Keystores / HSMs

Figure 12: Using Solaris KSSL Proxy for WebLogic SSL

Configuring KSSL for WebLogic SSL Acceleration

Using the KSSL kernel module as an SSL proxy requires obtaining and installing a certificate from a Certificate Authority[1]. Here are the steps to configure a testing KSSL accelerator:

Using OpenSSL Certificates (Flat-file Keystore)

1. Create a self-signed certificate using `openssl` utility

```
# /usr/sfw/bin/openssl req -x509 -nodes -days 365
           -subj  /C=US/ST=State/L=City/CN=serverhostname
               -newkey rsa:1024 -keyout  /etc/pki/key00.pem
                  -out  /etc/pki/cert00.pem
```

2. Concatenate and place all certificate artifacts in a single file.

```
# cat cert00.pem key00.pem > /etc/pki/mySSLCerts.pem
```

3. Move to the `/etc/pki` directory and execute the following command

```
# chown 600 mySSLCerts.pem
```

4. Configure the KSSL proxy and its redirect HTTP cleartext port. Assuming the Oracle WebLogic server default listen port, or cleartext port, is port 7001. Make sure the `/etc/pki/passwordfile` includes the password of the keystore.

```
# ksslcfg create -f pem -i /etc/pki/mySSLCerts.pem
           -x 7001 –p /etc/pki/passwordfile serverhostname 443
```

[1] For production deployments the use of a Certificate Authority is essential. However, a self-signed certificate can also be used for testing purposes.

5. Use the Service Management Facility (SMF) to verify that KSSL service is enabled.

   ```
   # svcs -a | grep "kssl"
   ```

6. Alternatively, use the Solaris 'netstat –an' command to verify KSSL is listening on port 443.

   ```
   # netstat -an | grep 443
   ```

7. Use a Web browser to check that the Oracle WebLogic server listens to the KSSL secured port. Go to `https://myservername.com:443`

Using PKCS#11 based Keystores and Hardware Security Modules

To ensure the security of private-key and server certificates and tamper-proof keystores, it is often recommended to use Hardware Security Modules (HSM). KSSL supports usage of PKCS#11 based HSMs (ex. Sun Crypto Accelerator 6000 PCIe Card), Software keystores (For ex. NSS, Solaris PKCS#11 softtoken). The following command and options are typically used for configuring PKCS#11 based keystores.

To configure KSSL using a Solaris PKCS#11 softtoken the steps are as follows:

1. Configure a "Sun Software PKCS#11 Softtoken" keystore using `pktool` utility.

   ```
   # pktool setpin keystore=pkcs11
   ```

2. Create a self-signed certificate

   ```
   # pktool  gencert  keystore=pkcs11  label="ksslCert"
           subject="C=US,O=Oracle, OU=ISVE, CN=serverhostname"
                         serial=0x000000001
   ```

3. Enable "Sun Software PKCS#11 Softtoken" token as metaslot

   ```
   # cryptoadm enable metaslot
                   token="Sun Software PKCS#11 Softtoken"
   ```

4. Configure the KSSL service identifying the PKCS#11 keystore assuming the Softtoken is created in the home directory of the user and the certificate alias is `ksslCert`. Make sure the `/etc/pki/passwordfile` includes the password of the keystore.

   ```
   # ksslcfg create –f pkcs11 –d $HOME/.sunw
         -T "Sun Software PKCS#11 Softtoken"
               -C "ksslCert"
                 -p /etc/pki/passwordfile
                        -x 7001 serverhostname  443
   ```

5. To verify that KSSL service is enabled, execute the `svcs` command as follows:

   ```
   # svcs -a | grep "kssl"
   ```

To configure KSSL using Sun Crypto Accelerator 6000 PCIe card as a HSM keystore - the steps are as follows:

1. Configure and verify that the Sun Crypto Accelerator 6000 PCIe card is initialized for use as a HSM. Refer to *Sun Cryptographic Accelerator 6000 PCIe Card Version 1.1 Documentation* for installation and configuration.

    a. The card is configured to use in FIPS mode and with a Primary Security Officer.
    b. A keystore is made available for use as a PKCS#11 token
    c. Create self-signed certificate using pktool or import SSL certificates (in PKCS12 format) obtained from a certificate authority.

2. Enable the newly created keystore as a metaslot.

```
# cryptoadm enable metaslot
                token="sca-keystore"
```

3. Configure the KSSL service identifying the PKCS#11 keystore assuming the password is stored in a file and the certificate alias is `ksslCert`. Make sure the `/etc/pki/passwordfile` includes the password of the keystore.

```
# ksslcfg create –f pkcs11
      -T "Sun Metaslot"
            -C "ksslCert"
                  -p /etc/pki/passwordfile
                        -x 7001 serverhostname  443
```

4. To verify that KSSL service is enabled, use the `svcs` command as follows:

```
# svcs -a | grep "kssl"
```

Using SSL Cipher Suites

To enforce the KSSL service negotiates with the end-user client (ex. Web browser) using specific SSL3/TLSv1 ciphersuites, the `ksslconfig` command must use `-c` option followed by the required list of `<ciphersuites>` in a sorted order. For example:

```
# ksslcfg create –c rsa_3des_ede_cbc_sha,rsa_des_cbc_sha
           -f pem -i /etc/pki/mySSLCerts.pem
            -x 7001 –p /etc/pki/passwordfile serverhostname 443
```

Using Apache Web Server SSL

Solaris bundled Apache Web Server supports Oracle SPARC T4 hardware-assisted cryptographic acceleration for SSL/TLS operations using PKCS#11 based OpenSSL Solaris bundled OpenSSL provides PKCS11 engine implementation to support delegation of cryptographic operations to hardware. You would able to verify the existence of PKCS11 engine by executing the following command:

```
# openssl engine
```

```
(t4) SPARC T4 engine support
(dynamic) Dynamic engine loading support
(pkcs11) PKCS #11 engine support
```

To begin with, make sure you complete the SSL configuration pre-requisites for the Apache Web server such as configuring the SSL certificates, SSL listen port etc. Once the configuration is verified for use, edit the ssl.conf file to include the following directive:

```
SSLCryptoDevice pkcs11
```

Restart the Web server. Now the cryptographic operations of the SSL protocol will automatically take advantage of the Oracle SPARC T4 hardware-assisted cryptographic acceleration.

Message-Layer Security Acceleration

The Oracle Web Services Manager (Oracle WSM) as part of Oracle Fusion Middleware plays the vital role in configuring and deploying message-layer security for SOA and XML Web services applications. The availability of SunPKCS11 and Oracle Ucrypto providers in Java Cryptographic Extension (JCE) framework enables the Oracle WSM to take advantage of Oracle SPARC T4 hardware-assisted cryptographic acceleration by off-loading WS-Security based cryptographic workloads.

Accelerating Message-Layer Security using Oracle WSM

The following steps explain how to configure Oracle WSM for message-layer security acceleration using the on-core cryptographic acceleration capabilities of Oracle SPARC T4 processor based servers. As a pre-requisite, it is assumed that Oracle WSM is installed and pre-configured with installation Oracle Fusion Middleware – SOA suite

1. To begin with, we will create and use a JKS keystore for supporting the initial storage of keys and certificates used with Oracle WSM. The steps for creating the JKS keystore and the key pairs for use with Oracle WSM is as follows:

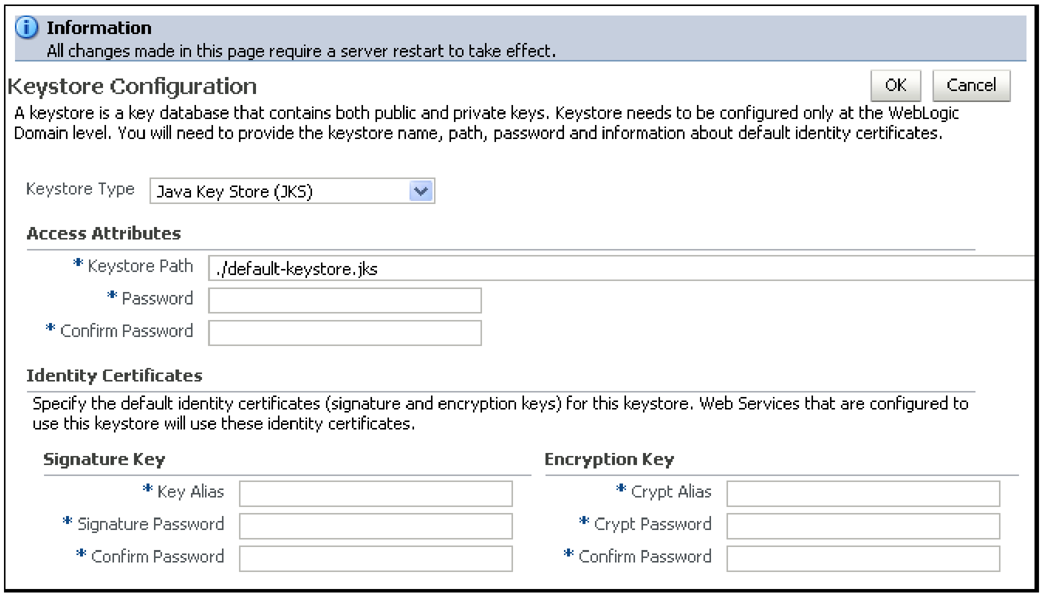
    - Create a new RSA key pair (private and public keys) for use with Oracle WSM to support signing and encrypting SOAP messages. Use the Java keytool with -genkey option to create a new RSA key pair with RSA-SHA1 as the signature algorithm. The following example shows creating a Java keystore `default-keystore.jks` and a key pair with alias `orakey` using key algorithm `RSA` and signature `SHA1withRSA`.

```
keytool -genkey -alias orakey -keyalg RSA
            -sigalg SHA1withRSA -dname
                     "CN=orakey, OU=Testing, C=US"
                          -keystore  default-keystore.jks
```

In case of production deployments, it is strongly suggested to acquire the Private/Public key pair and its certificate from a Certificate authority.

2. Configure Oracle WSM to use the newly created keystore. Here are the steps to configure the keystore for use with Oracle WSM:

   - Login to **`Enterprise Manager`**, which is available by default as it is installed with Oracle Fusion Middleware suite at http://localhost:7001/em
   - In the navigator pane, expand **`WebLogic Domain`** to show the domain for which you need to configure the keystore. Select the domain.
   - Using Fusion Middleware Control, click **`Weblogic Domain`**, then **`Security`**, and then **`Security Provider Configuration`**.
   - Click the plus sign (+) to expand the **`Keystore`** control near the bottom of the page, then click **`Configure`**. The Web Services Manager Keystore Configuration page is displayed, as shown in (Figure 13).


Information
All changes made in this page require a server restart to take effect.

Keystore Configuration

A keystore is a key database that contains both public and private keys. Keystore needs to be configured only at the WebLogic Domain level. You will need to provide the keystore name, path, password and information about default identity certificates.

Keystore Type: Java Key Store (JKS)

Access Attributes

* Keystore Path	./default-keystore.jks
* Password	
* Confirm Password	

Identity Certificates

Specify the default identity certificates (signature and encryption keys) for this keystore. Web Services that are configured to use this keystore will use these identity certificates.

Signature Key

* Key Alias	
* Signature Password	
* Confirm Password	

Encryption Key

* Crypt Alias	
* Crypt Password	
* Confirm Password	

Figure 13: Web Services Manager Keystore Configuration

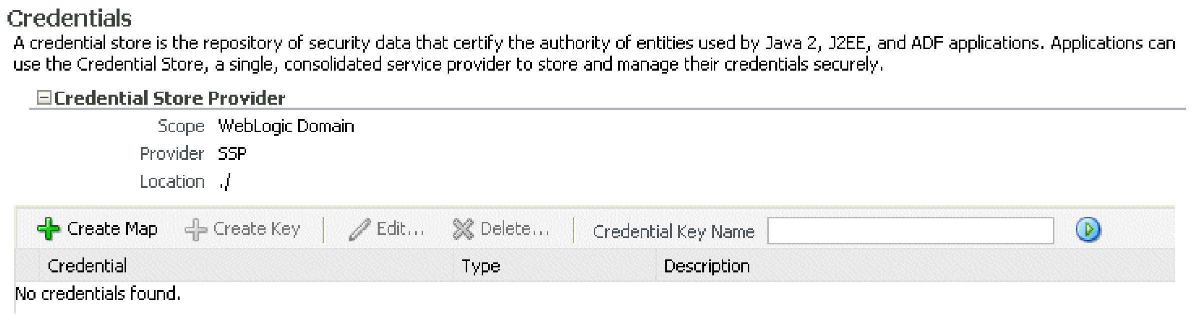
   - Select Keystore Type as JKS. Enter the path and name of the keystore - `default-keystore.jks` created for use with Oracle WSM. Also, enter the password for the keystore and re-confirm it.

- Next, enter the alias of the signature and encryption keys and the corresponding key passwords. It is important to the alias and password for the signature and encryption keys define the string alias and passwords used to store and retrieve the keys.
- Click on OK to submit and save the changes. Restart the SOA server and then relaunch the Oracle Enterprise Manager Fusion Middleware Control to take effect on the changes.

3. Optionally, configure an application-specific credential store provider to support storing, retrieving, and deleting credentials intended for Web services and other applications. For example, if you are using any username token policy, it is required to configure `csf-key` property, with a default value of `basic.credentials`. Oracle WSM uses this key to create and retrieve username/password token. The following steps are required to create a credential store key:

- Login to **`Enterprise Manager`**. In the navigator pane, expand **`WebLogic Domain`** to show the domain for which you need to configure the credential store. Select the domain.

- Using Fusion Middleware Control, click **`Weblogic Domain`**, then **`Security`**, and then **`Credentials`**. The Credential Store Provider Configuration page is displayed, as shown in Figure 14.


Credentials

A credential store is the repository of security data that certify the authority of entities used by Java 2, J2EE, and ADF applications. Applications can use the Credential Store, a single, consolidated service provider to store and manage their credentials securely.

Credential Store Provider

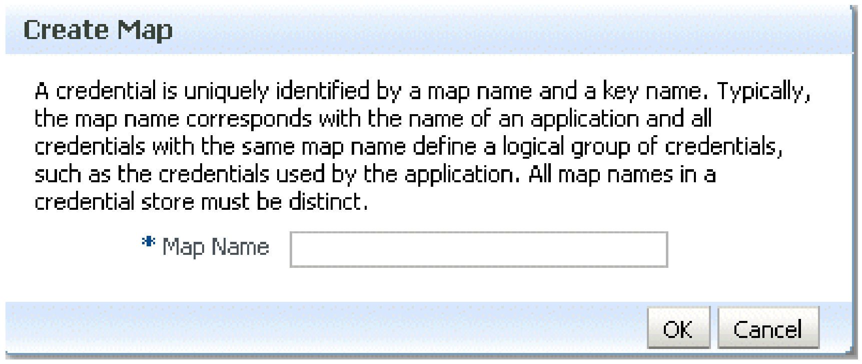
Scope	WebLogic Domain
Provider	SSP
Location	./

Credential	Type	Description

No credentials found.

Figure 14: Credential Store Provider Configuration Page

- Click `Create Map` and enter the map name as `oracle.wsm.security`, as shown in Figure 15. If this map is already created, then just skip this step.

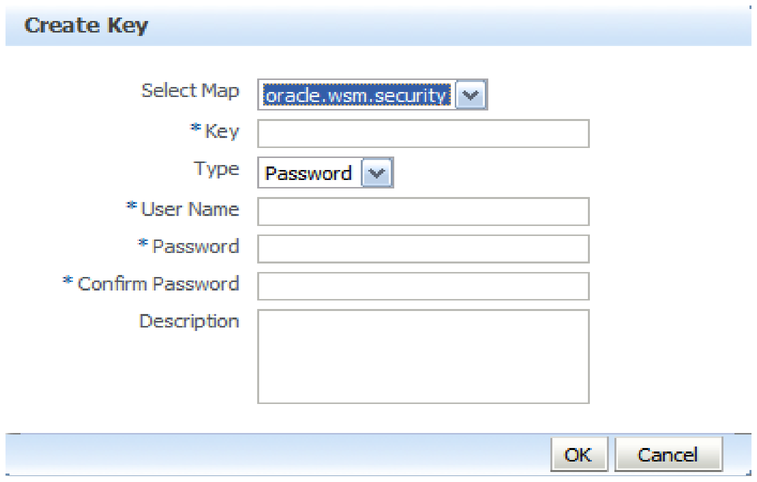

Create Map

A credential is uniquely identified by a map name and a key name. Typically, the map name corresponds with the name of an application and all credentials with the same map name define a logical group of credentials, such as the credentials used by the application. All map names in a credential store must be distinct.

* Map Name

Figure 15: Create Map Dialog

- Next, Click `Create key` to launch the Create Key dialog as shown in Figure 16.


Create Key

Select Map	oracle.wsm.security
* Key	
Type	Password
* User Name	
* Password	
* Confirm Password	
Description	

Figure 16: Create Key Dialog

- Select the map name `oracle.wsm.security`, if it is not already selected, enter the key name - `basic.credentials`.

- Select the key type, either `Password` or `Generic`. A password credential can store a username and password. A generic credential can store any other credential object. For a password credential, enter the username and password. Click OK.

- The above configuration is intended to support JKS keystore for verifying Oracle WSM setup. To enable cryptographic acceleration of Oracle WSM, it is required to migrate the keys to a PKCS#11 keystore.

- To verify the configuration using JKS keystore, it is recommended to test the configuration by deploying a sample Web service and attaching a OWSM policy. Refer to Steps 8 and 9 for test-driving a sample Web services and attaching policies.

4. It is quite critical to create PKCS#11 based keystore, before migrating the keys. Solaris 10 supports creating PKCS#11 keystore "Sun Software PKCS#11 Sofftoken" and NSS.

- Configure a "Sun Software PKCS#11 Sofftoken" keystore using SCF `pktool` utility. Set the PIN/Passphrase for access the

```
# pktool setpin keystore=pkcs11
  Create new passphrase:
  Re-enter new passphrase:
```

- Enable "Sun Software PKCS#11 Sofftoken" token as metaslot

```
# cryptoadm enable metaslot
    token="Sun Software PKCS#11 Sofftoken"
```

- Make sure the "Sun Software PKCS#11 Sofftoken" is available as a metaslot and available to access using Java keytool. If it is prompted, enter the passphrase of "Sun Software PKCS#11 Sofftoken".

```
# cryptoadm list –v metaslot

# keytool -list -storetype pkcs11 –keystore NONE
```

5. Using the keytool to migrating the keys from the JKS keystore to "Sun Software PKCS#11 Sofftoken".

```
# keytool –importkeystore
      -srckeystore /opt/Oracle/Middleware/default-keystore.jks
       -destkeystore NONE -srcstoretype JKS
         -deststoretype PKCS11
           -srcstorepass changeme -deststorepass scfpassword
```

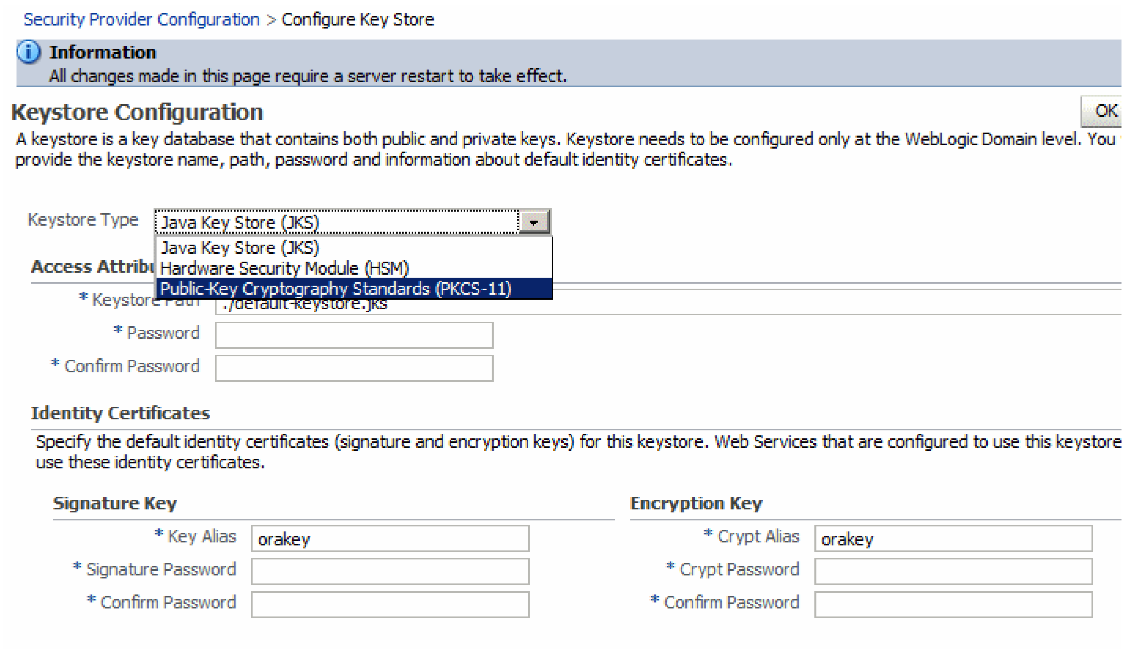
6. Verify whether the keys are present in PKCS11/HSM are accessible via keytool.

```
#  keytool -list -storetype pkcs11 -keystore NONE

Enter keystore password:
Keystore type: PKCS11
Keystore provider: SunPKCS11-Solaris

Your keystore contains 1 entries
```

```
SecretKeyEntry,
orakey, PrivateKeyEntry,
Certificate fingerprint (MD5):
EC:75:AC:21:C5:6F:6C:B5:49:2A:75:81:BB:D1:49:94
```

7. Update the PKCS#11 keystore "Sun Software PKCS#11 Softtoken" password in `Enterprise Manager` and change the keystore type to PKCS11.

   - To update PKCS11 keystore password using `Enterprise Manager`, follow the steps discussed in Step 2
   - In the Keystore Type drop-down, select Public Key Cryptographic Standards (PKCS#11) as shown in figure 17.


Security Provider Configuration > Configure Key Store

Information
All changes made in this page require a server restart to take effect.

Keystore Configuration
A keystore is a key database that contains both public and private keys. Keystore needs to be configured only at the WebLogic Domain level. You provide the keystore name, path, password and information about default identity certificates.

Keystore Type: Java Key Store (JKS)
- Java Key Store (JKS)
- Hardware Security Module (HSM)
- Public-Key Cryptography Standards (PKCS-11)

Access Attribu
* Keystore Path: ./default-keystore.jks
* Password:
* Confirm Password:

Identity Certificates
Specify the default identity certificates (signature and encryption keys) for this keystore. Web Services that are configured to use this keystore use these identity certificates.

Signature Key
* Key Alias: orakey
* Signature Password:
* Confirm Password:

Encryption Key
* Crypt Alias: orakey
* Crypt Password:
* Confirm Password:

Figure 17: Configuring PKCS#11 Keystore

- Enter the "Sun Software PKCS#11 Softtoken" password.
- Next, enter the signature and encryption key alias and passwords
- Click on OK and finally, restart the SOA server.
   a. Finally, restart the SOA server.

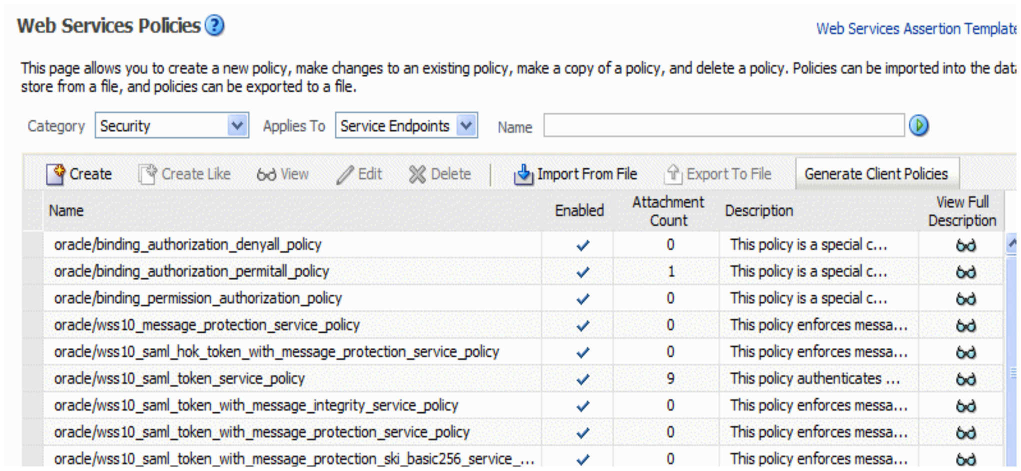
8. To verify the setup, deploy an XML Web service application. Refer to documentation - Deploying Web Services Applications.
   - http://download.oracle.com/docs/cd/E12839_01/web.1111/b32511/deploy.htm

9. To secure the Web service, assign the deployed XML Web service with Oracle WSM WS-Security Policies. To verify and test the configuration using PKCS11 keystore and enabling cryptographic

acceleration, we need assign any, policy which supports signing and encryption. Signing and encryption policies are identified with "`*message_protection*`" (For example: "`wss10_message_protection_service_policy`").

- Go to the Enterprise Manager navigator pane, expand **`WebLogic Domain`** to show the domain for which you want to see the policies. Select the domain.

- Using Fusion Middleware Control, click **`WebLogic Domain`**, then **`Web Services`** and then **`Policies`**. The Web Services Policies page is displayed as follows (Figure 18):



Web Services Policies

Web Services Assertion Templat

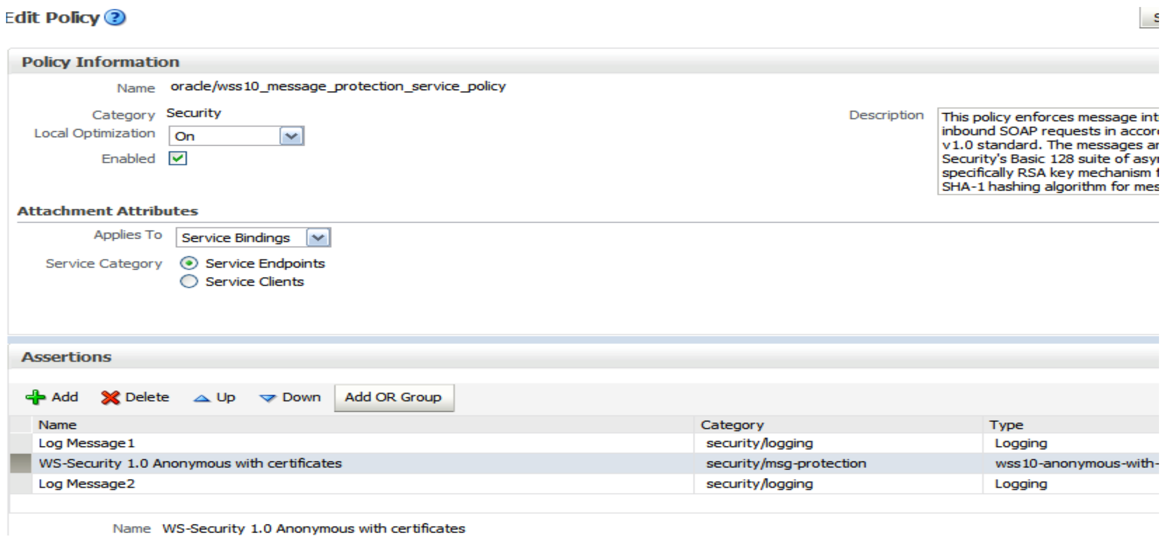
This page allows you to create a new policy, make changes to an existing policy, make a copy of a policy, and delete a policy. Policies can be imported into the dat store from a file, and policies can be exported to a file.

Category: Security | Applies To: Service Endpoints | Name:

Name	Enabled	Attachment Count	Description	View Full Description
oracle/binding_authorization_denyall_policy	✓	0	This policy is a special c...	👓
oracle/binding_authorization_permitall_policy	✓	1	This policy is a special c...	👓
oracle/binding_permission_authorization_policy	✓	0	This policy is a special c...	👓
oracle/wss10_message_protection_service_policy	✓	0	This policy enforces messa...	👓
oracle/wss10_saml_hok_token_with_message_protection_service_policy	✓	0	This policy enforces messa...	👓
oracle/wss10_saml_token_service_policy	✓	9	This policy authenticates ...	👓
oracle/wss10_saml_token_with_message_integrity_service_policy	✓	0	This policy enforces messa...	👓
oracle/wss10_saml_token_with_message_protection_service_policy	✓	0	This policy enforces messa...	👓
oracle/wss10_saml_token_with_message_protection_ski_basic256_service_...	✓	0	This policy enforces messa...	👓

Figure 18: Available Web Services Policies

- To attach a select security policy to a deployed Web Service, Refer to documentation: [Attaching Policies to Web Services](http://download.oracle.com/docs/cd/E12839_01/web.1111/b32511/attaching.htm).
    - o http://download.oracle.com/docs/cd/E12839_01/web.1111/b32511/attaching.htm
- To edit the OWSM policy and change the default algorithm suite (Default is `Basic128`) to use a different algorithm (ex. `Basic256Rsa15` refers to AES-256 and Rsa15 key wrap), the steps are follows:
    - o Navigate to the `Web Services Policy` page, as described in section 7 of Attaching Policies to Web Services.
    - o From the `Web Services Policies` page, select the attached policy from the `Policies` table and click Edit (in order to change algorithm suite, you should select any of the message protection policy).
    - o On the `Edit Policy` page (Refer Figure 19), in `Assertions` section select assertion that has `msg-protection` category. Generally such policy is selected by default.


Edit Policy

Policy Information

Name oracle/wss10_message_protection_service_policy
Category Security
Local Optimization On
Enabled ✓
Description This policy enforces message int inbound SOAP requests in accor v1.0 standard. The messages ar Security's Basic 128 suite of asy specifically RSA key mechanism f SHA-1 hashing algorithm for mes

Attachment Attributes

Applies To Service Bindings
Service Category [● Service Endpoints] ○ Service Clients

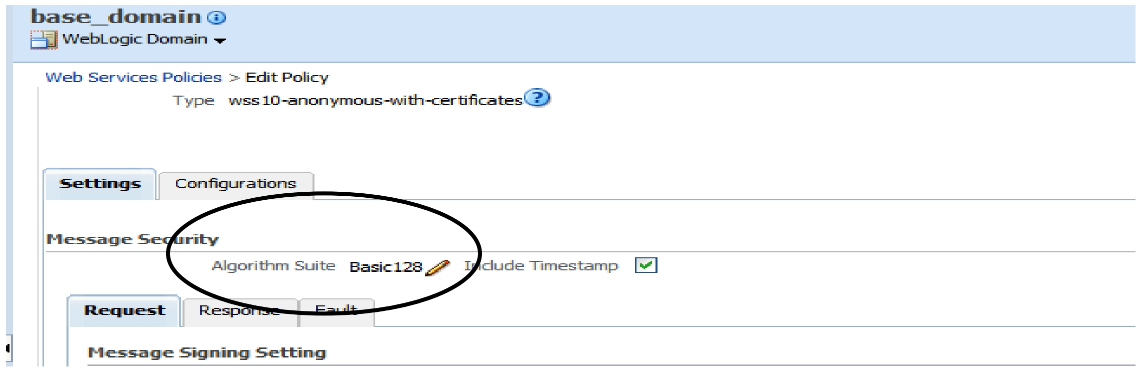
Assertions

Name	Category	Type
Log Message1	security/logging	Logging
WS-Security 1.0 Anonymous with certificates	security/msg-protection	wss10-anonymous-with-
Log Message2	security/logging	Logging

Name WS-Security 1.0 Anonymous with certificates

Figure 19: Editing Web Services Policies

- Click on the `Settings` tab, you will find Algorithm Suite as `Basic128` (Refer Figure 20).


base_domain
WebLogic Domain

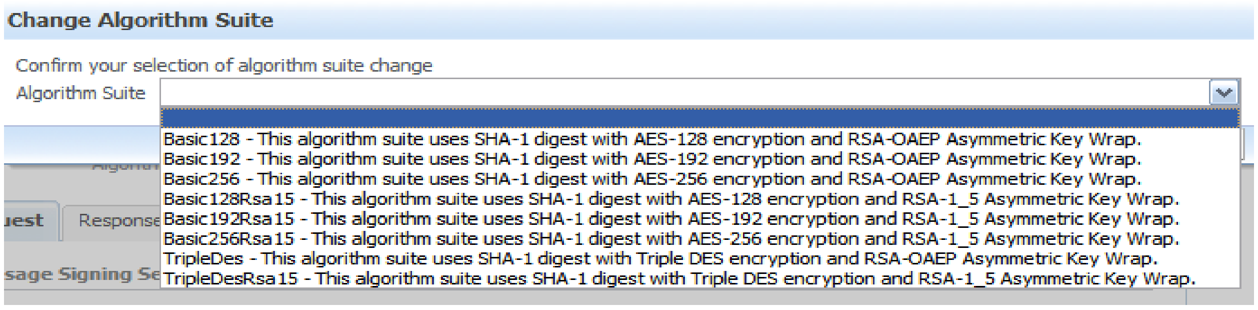
Web Services Policies > Edit Policy
Type wss10-anonymous-with-certificates

Message Security
Algorithm Suite Basic128 Include Timestamp ✓

Message Signing Setting

Figure 20: Changing the Algorithm Suite

- Click on the pencil sign on the Algorithm Suite, `Change Algorithm Suite` page will appear. Select the desired algorithm (Figure 21) and Click Save. The changes will take effect at the next polling interval for policy changes. If you are using a database-based metadata repository, each time you save a change to your policy, a new version is created, and the older versions are retained.



Basic128 - This algorithm suite uses SHA-1 digest with AES-128 encryption and RSA-OAEP Asymmetric Key Wrap.
Basic192 - This algorithm suite uses SHA-1 digest with AES-192 encryption and RSA-OAEP Asymmetric Key Wrap.
Basic256 - This algorithm suite uses SHA-1 digest with AES-256 encryption and RSA-OAEP Asymmetric Key Wrap.
Basic128Rsa15 - This algorithm suite uses SHA-1 digest with AES-128 encryption and RSA-1_5 Asymmetric Key Wrap.
Basic192Rsa15 - This algorithm suite uses SHA-1 digest with AES-192 encryption and RSA-1_5 Asymmetric Key Wrap.
Basic256Rsa15 - This algorithm suite uses SHA-1 digest with AES-256 encryption and RSA-1_5 Asymmetric Key Wrap.
TripleDes - This algorithm suite uses SHA-1 digest with Triple DES encryption and RSA-OAEP Asymmetric Key Wrap.
TripleDesRsa15 - This algorithm suite uses SHA-1 digest with Triple DES encryption and RSA-1_5 Asymmetric Key Wrap.

Figure 21: Changing the Algorithm Suite

10. To enable offloading to hardware and enable cryptographic acceleration, make sure the underlying Java runtime environment's security property file `java.security` is configured to use SunPKCS11 or Oracle Ucrypto provider. In case of using SunPKCS11 provider, make sure to include the cryptographic mechanisms specified in the WS-SecurityPolicy algorithm suite is include in the list of `enabledMechanisms` or by removing them from the list of `disabledMechanisms` of the Java SunPKCS11 configuration file. If the specified algorithm suite is `Basic256Rsa15`, it uses `Aes256` encryption and `Rsa-oaep-mgf1p` for key wrap. To enable acceleration, you need to remove the required bulk encryption algorithms in the `disabledMechanisms` list that forces those operations (ex. `CKM_AES`).

11. Restart the Fusion Middleware Control.

Verifying Hardware-Assisted Security for Oracle Fusion Middleware

To ensure the hardware-assisted cryptographic acceleration is configured to use and working with the security scenarios, it is recommended to use the following Solaris DTrace script.

```
#!/usr/sbin/dtrace -s
pid$1:libsoftcrypto:yf*:entry,
pid$1:libmd:yf*:entry
{
 @[probefunc] = count();
}
tick-10sec
{
 printa(@);
 clear(@);
 trunc(@,0);
}
tick-100sec
{exit(0);}
```

Save the above script as 'cryptoverify.d' file and run the Dtrace script including the "Weblogic server's Java process id" as command line argument.

# dtrace -s cryptoverify.d <WeblogicServer Process ID>

For example, in an XML encryption scenario using AES algorithm, a positive and growing value of aes jobs indicates that cryptographic acceleration is operational on the target AES bulk encryption payloads – Refer to the following sample output.

```
# dtrace -s cryptoverify.d 5774
dtrace: script 'crypto-t4.d' matched 51 probes
CPU     ID                    FUNCTION:NAME
 65  83719                      :tick-10sec
  yf_aes128_ecb_decrypt                                      39922
  yf_aes128_load_keys_for_decrypt                            39922

 65  83719                      :tick-10sec
  yf_aes128_ecb_decrypt                                      44108
  yf_aes128_load_keys_for_decrypt                            44108

 65  83719                      :tick-10sec
  yf_aes128_ecb_decrypt                                      44534
  yf_aes128_load_keys_for_decrypt                            44534
..
```

Oracle Database Security Using Oracle SPARC T4 Processor

Oracle database ensures confidentiality and integrity of data in transit and at rest using encryption at all levels. As part of Oracle Advanced Security options the Oracle 11g Transparent Data Encryption (TDE) features support for network encryption, tablespace encryption and column-level data encryption. TDE uses standard algorithms and facilitates a built-in key management services for supporting data encryption. Since Oracle Database 11g (11.2.03), TDE extended support for hardware-assisted cryptographic acceleration using Oracle SPARC T4 processor and Solaris 11to support offloading cryptographic processing associated with tablespace encryption, column-level encryption, network encryption and master key based operations (Refer: Figure 22).

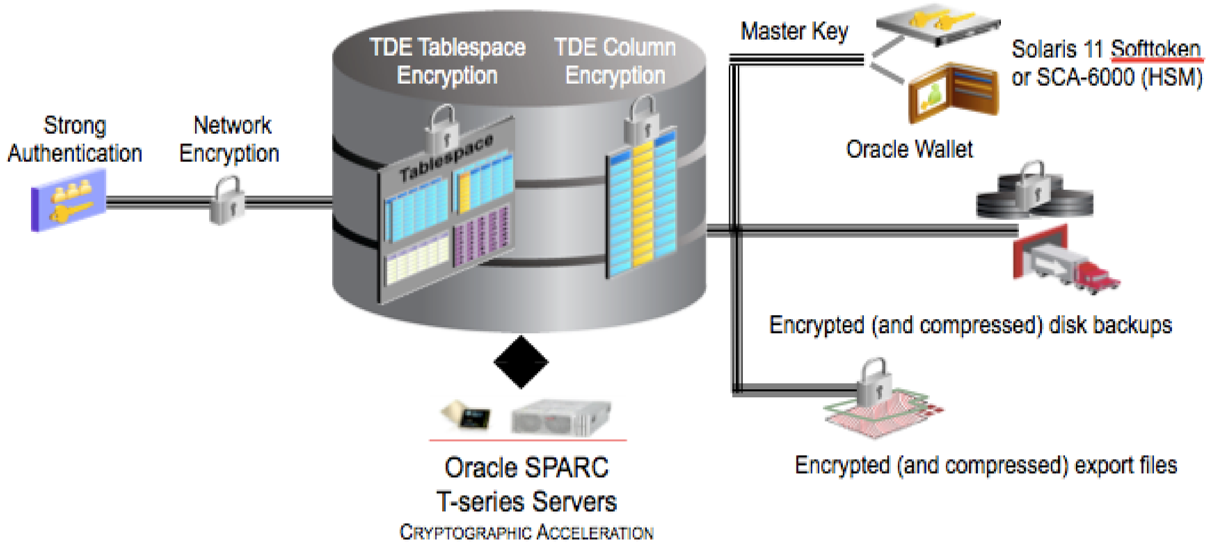

TDE Tablespace Encryption
TDE Column Encryption
Tablespace
Strong Authentication
Network Encryption
Master Key
Solaris 11 Softtoken or SCA-6000 (HSM)
Oracle Wallet
Encrypted (and compressed) disk backups
Encrypted (and compressed) export files
Oracle SPARC T-series Servers
CRYPTOGRAPHIC ACCELERATION

Figure 22: Oracle Database Security: Applied Security Scenarios using Oracle SPARC T4 processor

Transparent Data Encryption (TDE): Applied Scenarios

TDE plays the role of encryption and decryption of data stored in an Oracle database and in transit by providing support for all encryption operation applied to network communication, tablespace and column-level encryption and encrypted backups. TDE has been tested and verified to use Oracle SPARC T4 hardware assisted cryptographic acceleration for all encryption operations. The applied security scenarios are as follows:

- Tablespace encryption
- Network encryption
- Master Key Management
    - Solaris PKCS#11 Softtoken based Oracle wallet
        - Centralized key store for securing the master key used to encrypt and decrypt the keys performing actual data encryption.
            - Encryption/decryption of tablespace and column encryption keys.
            - Encryption/decryption support for Oracle Data Pump utility.
            - Encryption/decryption of backup/restore using Oracle Recovery Manager (RMAN)
    - Master key backup and recovery

Master key Management using Solaris PKCS#11 Softtoken

Oracle 11g support the use PKCS#11 based HSM keystore as Oracle Wallet. Using Solaris PKCS#11 softtoken based Oracle Wallet secures the master key from duplication and copying during database and filesystem backups. This can be done using Solaris PKCS#11 Sofftoken referred to as "Sun Software PKCS#11 Sofftoken".

1. Configure a "Sun Software PKCS#11 Sofftoken" keystore using SCF `pktool` utility. Set the PIN/Passphrase for access the Softtoken key store.

```
# pktool setpin keystore=pkcs11
  Create new passphrase:
  Re-enter new passphrase:
```

2. Enable "Sun Software PKCS#11 Sofftoken" token as metaslot

```
# cryptoadm enable metaslot
    token=“Sun Software PKCS#11 Sofftoken”
```

- Copy the PKCS#11 library to Oracle suggested directory structure. On Solaris SPARC environment, create a directory for the PKCS#11 library:

```
mkdir –p /opt/oracle/extapi/64/hsm/sun/1.0.0/lib
```

Copy Solaris libpkcs11.so to the PKCS#11 library directory

```
copy /usr/lib/sparcv9/libpkcs11.so
       /opt/oracle/extapi/64/hsm/sun/1.0.0/lib
```

3. Make sure that Oracle install user:group for Oracle PKCS#11 library directory and it is assigned with read and write privileges.

```
chown –R oracle:oinstall  <..directory...>
```

- In the Solaris environment, you may choose to set an environment variable (in the Oracle default user shell) "SOFTTOKEN_DIR" .

```
export SOFTTOKEN_DIR=/export/home/oracle/.sunw
```

To configure TDE to use Solaris PKCS#11 Softtoken, initially it is required to setup an HSM based Oracle Wallet identifying the source of master key as HSM.

- Edit the $TNS_ADMIN/sqlnet.ora and add an ENCRYPTION_WALLET_LOCATION parameter as follows:

```
ENCRYPTION_WALLET_LOCATION =
  (SOURCE=(METHOD=HSM) (METHOD_DATA=
(DIRECTORY = /export/home/oracle/11g/network/admin/)))]
```

- Log in to SQLPlus as "system" or "sysdba" and create a HSM Wallet.

  ```
  $ sqlplus "/ as sysdba"

  SQL> alter system set encryption key

                    identified by "HSM Password";
  ```

  `Username:Password` are the credentials of the dedicated user account for TDE for support performing master key management operations with the Solaris PKCS#11 softtoken keystore.

If the database is previously using Oracle software wallet, then you would able to migrate the master key to the configured Solaris PKCS#11 Softtoken. The migration process automatically decrypts existing data objects and re-encrypts them using the newly created master encryption key on Solaris PKCS#11 Softtoken. In case of Oracle TDE configuration previously configured using a "software wallet" then it is required to migrate the master key from the software wallet to HSM by adding `MIGRATE USING "software_wallet_password"` clause to the preceding sqlplus command. The `software_wallet_password` is the original password for the software wallet.

```
SQL> alter system set encryption key identified by

          "HSM Password" migrate using

                        "software_wallet_password";
```

Accelerating Tablespace Encryption

Oracle Database 11.2.0.3 introduced support for Oracle SPARC T4 and T3 hardware-assisted cryptographic acceleration for Oracle TDE. As the installation process automatically identifies the processor of the host machine. Technically there is no setup required for the use of Oracle SPARC T3/T4 hardware accelerated cryptography.

Once it is deployed, Oracle database will use Oracle SPARC T4 hardware accelerated cryptography for both encryption and decryption operations involved with tablespace encryption, network encryption, encrypted backups and restore and encrypted dump files.

Testing and Verifying Oracle TDE on Oracle SPARC T4 and Solaris 11

To test and verify TDE using the master encryption key stored in Solaris PKCS#11 Softtoken. The following is a list of SQL examples showing TDE operations that relies on the Solaris PKCS#11 Softtoken resident master key.

1. Make sure the database is up and running. Log in and connect to *sqlplus* as *system*.

   ```
   $ sqlplus "/ as sysdba"
   SQL> startup;
   SQL> connect as system/password;
   ```

2. Verify opening and closing the Solaris PKCS#11 Softtoken based HSM wallet. Make sure you use the username and password created for accessing TDE.

```
SQL> ALTER SYSTEM SET WALLET OPEN IDENTIFIED BY "tdepassword";
System altered.
SQL> select WRL_TYPE, STATUS from v$encryption_wallet;
WRL_TYPE                  STATUS
--------------------      --------------
HSM                        OPEN
SQL> ALTER SYSTEM SET WALLET CLOSE IDENTIFIED BY   "tdepassword";
System altered.
```

3. Creating an encrypted tablespace
   - Make sure the HSM wallet is open.

```
SQL> ALTER SYSTEM SET WALLET OPEN IDENTIFIED BY "tdepassword";
```

   - Create an encrypted tablespace.

```
SQL> CREATE TABLESPACE SCASecuredTablespace
  2  DATAFILE '/export/home/oracle/11g/oradata/scasecuretbs1.dbf'
  3  SIZE 50M
  4  ENCRYPTION
  5  DEFAULT STORAGE(ENCRYPT);
```

4. Creating a Table on encrypted tablespace, which automatically encrypts all data objects stored.

```
SQL> CREATE TABLE PERSON
  2  (first_name VARCHAR2(11),
  3  last_name VARCHAR2(10),
  4  social_security_number NUMBER(9),
  5  address VARCHAR2(25),
  6  city VARCHAR2(25),
  7  state VARCHAR2(2)) TABLESPACE SecuredTablespace;
```

5. Encrypted export/import files using Oracle Data Pump utility can use HSM resident master key for encrypting and decrypting dump files.
   - By default, without specifying encryption all export dump file stored in unencrypted file. Here is the example showing export file dumped without encryption.

```
$ expdp system/oracle@sid tables=employee
```

- To enforce export dump file encryption using master key, first make sure the HSM wallet remains open. You need to use ENCRYPTION_MODE=TRANSPARENT to enable encryption of the dumpfile using the master key stored in the HSM wallet. Specifying option, ENCRYPTION_MODE=DUAL encrypts the dump set using the master key stored in the wallet and additionally using the password for encryption. Here is the example:

```
$ expdp system/oracle@sid tables=employee encryption=all
encryption_password=pwd4encrypt encryption_algorithm=AES256
encryption_mode=DUAL
```

- To import the dump file encrypted using master key, make sure the HSM wallet remains open and set the option specifying the password used for encryption. Here is an example:

```
$impdp system/oracle@Ssid encryption_password=pwd4encrypt
tables=employee table_exists_action=replace
```

6. Backup and Restore of database using Oracle Recovery Manager (RMAN) can use HSM resident master key.

- Make sure the HSM wallet is open before performing backup/restore/recover database command and also ensure the database is in archivelog mode. Here is the example:

```
SQL> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> startup mount;
ORACLE instance started.
Database mounted.
SQL> alter database archivelog;
Database altered.
SQL> alter database open;
Database altered.
SQL> ALTER SYSTEM SET WALLET OPEN IDENTIFIED BY "oracle:password";
System altered.
SQL> exit
```

- Use the rman utility, make sure to set encryption on before executing the backup command. Here is the example:

```
RMAN> connect target sysoper/oracle;

    connected to target database: sid (DBID=1555558107)

RMAN> set encryption on;


RMAN> backup as compressed backupset database;
```

Accelerating Oracle Network Data Encryption

Oracle Network data encryption allows to encrypt the data in transit over the network between the Oracle database server and the Oracle clients. Oracle database supports the use of Oracle SPARC T4 and T3 hardware-assisted cryptographic acceleration for Oracle network data encryption. To enable this support, the Oracle Wallet must be configured to Wallet type as PKCS#11 that allows the use of Solaris PKCS#11 Softtoken instead of filesystem based Wallet. Oracle Wallet Manager application allows configuring the PKCS#11 based wallet to support storing and managing PKI certificate credentials including private keys, certificates, and trusted certificates needed by SSL/TLS protocol for securing communication and client/server authentication.

As a pre-requisite, it is required to configure a "Sun Software PKCS#11 Sofftoken" keystore. Refer to steps (1) to (3) as described in section "Master Key Management using Solaris PKCS#11 Softtoken". Once configured, use the Oracle Wallet Manager (owm) utility to set up the PKCS#11 based Oracle Wallet. To configure SSL/TLS, refer to the steps described in section "Configuring Secure Sockets Layer Authentication" of Oracle Database Advanced Security Administrator's Guide. It is critical that the server choose SSL cipher suites including algorithms supported by the Oracle SPARC T4/T3 processor. For example, SSL_RSA_WITH_AES_128_CBC_SHA is supported as it uses RSA (Handshake and authentication), AES-128 for bulk encryption. The Oracle default SSL cipher suite SSL_RSA_WITH_RC4_128_SHA requires the use of RC4 for bulk encryption, which is not supported.

Verifying Hardware-Assisted Security for Oracle TDE

To ensure the hardware-assisted cryptographic acceleration is configured to use and working with the security scenarios of Oracle TDE, it is recommended to use the following Solaris DTrace script.

```
<<<<  dbcrypto.d>>>>>

#!/usr/sbin/dtrace -s

pid$1:libsoftcrypto:yf*:entry
 {
    @[probefunc] = count();

 }
  tick-10sec
 {
    printa(@);
    clear(@);
    trunc(@,0);
 }

<<<<  dbcrypto.d>>>>>
```

Save the above script as 'dbcrypto.d' file and run the Dtrace script including the "the process id of the Oracle foreground process that performs crypto operations" as command line argument.

```
   #./dbcrypto.d <PID of Oracle process>
```

For example, in an XML encryption scenario using AES algorithm, a positive and growing value of aes jobs indicates that cryptographic acceleration is operational on the target AES bulk encryption payloads of the database.

Securing Data at Rest using ZFS Encryption

Both Oracle Fusion Middleware and Database application are tested and verified to install and run on encrypted file system provided Solaris 11 ZFS encryption capabilities. By default, ZFS uses the Oracle Solaris 11 Cryptographic Services APIs, which automatically benefits from the hardware acceleration of AES algorithm available on the SPARC T-series processors. The policy for encryption is set at the dataset level when datasets (file systems or ZVOLs) are created. Each ZFS on disk block (smallest size is 512 bytes, largest is 128k) is encrypted using AES algorithm in either CCM or GCM mode. The wrapping keys need to be provided by the Solaris administrator who creates the firesystem, which can be changed at any time without taking the file system off line. The data encryption keys are randomly generated at dataset creation time. The easiest way to create the wrapping keys is to use the existing Solaris `pktool` command:

```
$ pktool genkey
       keystore=file keytype=aes keylen=128 outkey=/media/stick/mykey
```

Using ZFS encryption support can be as easy as this:

```
# zfs create -o encryption=on -o
         keysource=raw,file:///media/stick/mykey
                                             myfilesystem/cryptofs
```

For more details on ZFS encryption, refer to Oracle Solaris ZFS Administration Guide.

Summary

This whitepaper presented on the Oracle SPARC T4 processor's on-core cryptographic instruction features intended for hardware assisted cryptographic acceleration. The paper unveiled the core mechanisms, configuration, deployment strategies and the role and relevance of using the Solaris Cryptographic Framework and Java Cryptographic Extensions based techniques for delivering high-performance end-to-end security solution for Oracle Fusion Middleware and Oracle Database server applications. Adopting to SSL/TLS, WS-Security, encrypted data in transit and encrypted data at rest has become critical for delivering end-to-end security of multi-tier business applications and to meet regulatory compliance mandates.

The use of Oracle SPARC T4 hardware assisted cryptographic acceleration for end-to-end security deployments has certainly yielded tangible, immediate and cost-efficient results in the form of faster secure transactions and better response times – all without adding any additional security equipment costs, changes in power usage profiles or elaborate system configurations. Additionally, the performance results clarify the massive burden un-accelerated cryptographic workloads can have on a server. To summarize, the Oracle SPARC Enterprise T-Series servers and blades has proven demonstrating high performance security with consistent scalability for Oracle Weblogic server, Oracle Fusion Middleware and Oracle Database server deployed applications while also delivering reductions in space, power consumption, and cost.

Further References

- Oracle Fusion Middleware Documentation Library
    - http://docs.oracle.com/cd/E14571_01/soa.htm
- Oracle Database Advanced Security Administrator's guide
    - http://docs.oracle.com/cd/E11882_01/network.112/e10746/toc.htm
- Oracle Database Security guide
    - http://docs.oracle.com/cd/E11882_01/network.112/e16543/toc.htm
- Oracle SPARC Enterprise T-Series servers
    - http://www.oracle.com/us/products/servers-storage/servers/sparc-enterprise/t-series/index.html
- Oracle Cryptographic Accelerator 6000 PCIe Card Version 1.1 Documentation
    - http://www.oracle.com/us/products/servers-storage/networking/031146.htm
- Securing Oracle WebLogic Web Services
    - http://docs.oracle.com/cd/E24329_01/web.1211/e24488/toc.htm
- Java PKCS#11 Reference Guide
    - http://download.oracle.com/javase/6/docs/technotes/guides/security/p11guide.html
- Oracle Solaris 11 Administration: Security Services
    - http://docs.oracle.com/cd/E23824_01/html/821-1456/index.html
- Oracle Solaris 11 ZFS Administration Guide
    - http://docs.oracle.com/cd/E19253-01/819-5461/index.html

Oracle is committed to developing practices and products that help protect the environment

Oracle SPARC T4 Hardware-Assisted Security for Oracle Fusion Middleware and Oracle Database Applications,

April 2012, Version 1.0

Author: Ramesh Nagappan

Contributing Authors:
Nitin Handa
Sujeet Vasudevan
Ravindra Talashikar

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.
Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com